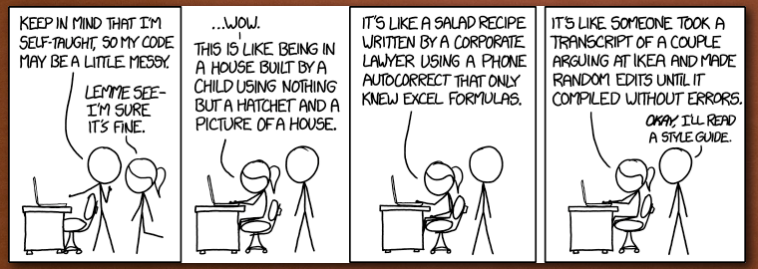
PSEUDOCODE



https://xkcd.com/1513/

LEARNING OBJECTIVES

- Explain the benefits of pseudocode over natural language or a programming language
- Represent algorithms using pseudocode

INTRODUCTION

- Algorithmic problem solving focuses on algorithms suitable for computers such as searching lists and matching patterns.
- Pseudocode is a tool for designing algorithms but does not run on a computing device.
- This chapter (2) uses a set of problems to illustrate algorithmic problem solving, including those with conditional statements and loops.

REPRESENTING ALGORITHMS (1 OF 5)

- Natural language is:
  - Expressive and easy to use
  - Verbose, unstructured, and ambiguous
- Programming languages are:
  - Structured and designed for computers
  - Grammatically fussy and cryptic
- **Pseudocode** lies somewhere between these two and is used to design algorithms prior to coding them

REPRESENTING ALGORITHMS NATURAL LANGUAGE (1 OF 2)

FIGURE 2.1

Initially, set the value of the variable carry to 0 and the value of the variable i to 0. When these initializations have been completed, begin looping as long as the value of the variable *i* is less than or equal to (*m* - 1). First, add together the values of the two digits a_i and b_i and the current value of the carry digit to get the result called c_i Now check the value of c_i to see whether it is greater than or equal to 10. If c_i is greater than or equal to 10, then reset the value of carry to 1 and reduce the value of c_i by 10; otherwise, set the value of carry to 0. When you are …

REPRESENTING ALGORITHMS NATURAL LANGUAGE (2 OF 2)

… finished with that operation, add 1 to i and begin the loop all over again. When the loop has completed execution, set the leftmost digit of the result c_m to the value of carry and print out the final result, which consists of the digits c_m c_{m-1}…c_0. After printing the result, the algorithm is finished, and it terminates.

The addition algorithm of Figure 1.2 expressed in natural language

What can we say about natural language?

- is it concise?
- is it unambiguous?
- does it convey structure?

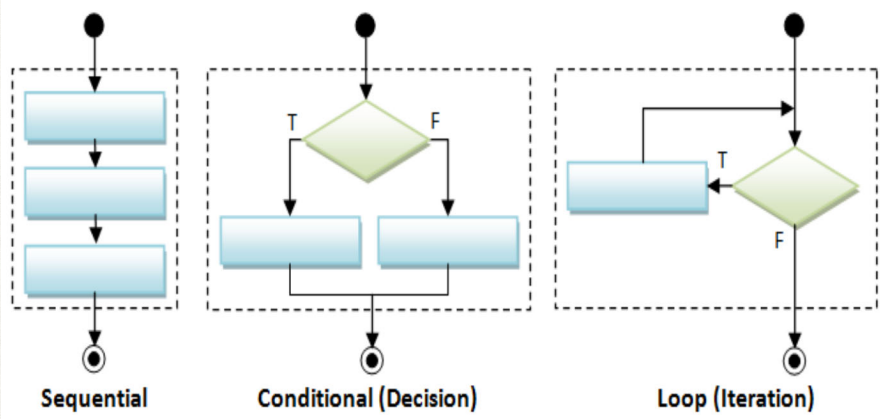
REPRESENTING ALGORITHMS PROGRAMMING LANGUAGE

FIGURE 2.2

```
{
Scanner inp = new Scanner(System.in);
int i, m, carry;
int[] a = new int[100];
int[] b = new int[100];
int[] c = new int[100];
m = inp.nextInt();
for (int j = 0;j <= m-1;j++) {
       a[j] = inp.nextInt();
       b[j] = inp.nextInt();
}
carry = 0;
i = 0;
while (i < m) {
    c[i] = a[i] + b[i] + carry;
    if (c[i] >= 10)
              .
              .
              .
```

The beginning of the addition algorithm of Figure 1.2 expressed in a high-level programming language

BASIC PSEUDOCODE CONSTRUCTS



REPRESENTING ALGORITHMS (2 OF 5)

- Sequential operations perform a single task
- The three basic sequential operations:
  - **Computation**: a single numeric or symbol calculation
  - **Input**: gets data values from outside the algorithm
  - **Output**: sends data values to the outside world
- A **sequential algorithm** is made up only of sequential operations
- A **variable** is a named storage location to hold a data value
- Example: computing average miles per gallon

REPRESENTING ALGORITHMS SEQUENTIAL ALGORITHM

FIGURE 2.3

Step	Operation
1	Get values for *gallons used, starting mileage, ending mileage*
2	Set value of *distance driven* to *(ending mileage - starting mileage)*
3	Set value of *average miles per gallon* to *(distance driven ÷ gallons used)*
4	Print the value of *average miles per gallon*
5	Stop

Algorithm for computing average miles per gallon (version 1)

SHORTER VERSION

- Because programmers are lazy, we don't like writing pseudocode which is too verbose.

```
get gallons used, starting mileage, ending mileage
distance driven ← ending mileage - starting mileage
average miles per gallon ← distance driven ÷ gallons used
print average miles per gallon ↵
```

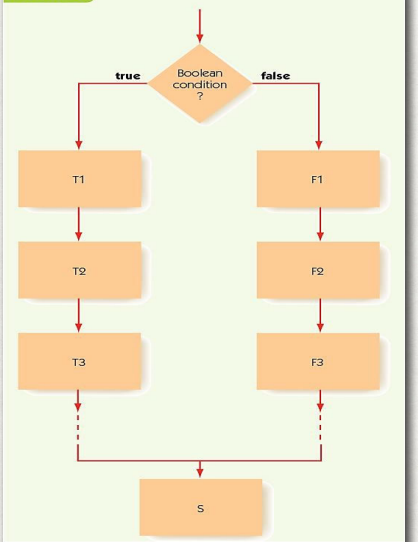
- N.B. Full names are still used for the data (in a programming language these will be variables)
- Why didn't I use an "=" for setting data values?
- Why did I use ↵?

REPRESENTING ALGORITHMS (3 OF 5)

- **Control operation:** changes the normal flow of control
- **Conditional statement**: asks a question and selects among alternative options:
  1. Evaluate the true/false condition
  2. If the condition is true, then do the first set of operations and skip the second set
  3. If the condition is false, skip the first set of operations and do the second set
- Example: check for good or bad gas mileage

REPRESENTING ALGORITHMS CONDITIONAL STATEMENT (1 OF 2)

FIGURE 2.4



The if/then/else pseudocode statement

REPRESENTING ALGORITHMS CONDITIONAL STATEMENT (2 OF 2)

FIGURE 2.5

Step	Operation
1	Get values for *gallons used, starting mileage, ending mileage*
2	Set value of *distance driven* to (*ending mileage – starting mileage*)
3	Set value of *average miles per gallon* to (*distance driven ÷ gallons used*)
4	Print the value of *average miles per gallon*
5	If *average miles per gallon* is >25.0 then
6	Print the message 'You are getting good gas mileage'
	Else
7	Print the message 'You are NOT getting good gas mileage'
8	Stop

Second version of the average miles per gallon algorithm

SHORTER VERSION

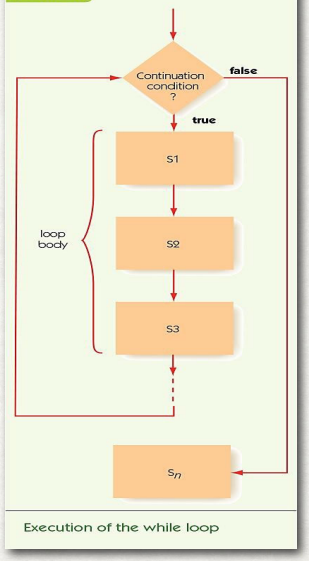
```
get gallons used, starting mileage, ending mileage
distance driven ← ending mileage - starting mileage
average miles per gallon ← distance driven ÷ gallons used
print average miles per gallon ↵
if average miles > 25 then
    print "You are getting good gas mileage." ↵
else
    print "You are NOT getting good gas mileage." ↵
end if
```


REPRESENTING ALGORITHMS (4 OF 5)

- **Iteration:** an operation that causes looping, repeating a block of instructions
- While statement repeats while a condition remains true
  - **Continuation condition**: a test to see if while loop should continue
  - **Loop body**: instructions to perform repeatedly
- Example: repeated mileage calculations

REPRESENTING ALGORITHMS ITERATION AND LOOP BODY (1 OF 2)

FIGURE 2.6



Execution of the while loop

REPRESENTING ALGORITHMS ITERATION AND LOOP BODY (2 OF 2)

FIGURE 2.7

something bad happening here

Step	Operation
1	*response* = Yes
2	While (*response* = Yes) do Steps 3 through 11
3	Get values for *gallons used*, *starting mileage*, *ending mileage*
4	Set value of *distance driven* to (*ending mileage* – *starting mileage*)
5	Set value of *average miles per gallon* to (*distance driven* ÷ *gallons used*)
6	Print the value of *average miles per gallon*
7	If average miles per gallon > 25.0 then
8	Print the message 'You are getting good gas mileage'
	Else
9	Print the message 'You are NOT getting good gas mileage'
10	Print the message 'Do you want to do this again? Enter Yes or No'
11	Get a new value for *response* from the user
12	Stop

Third version of the average miles per gallon algorithm

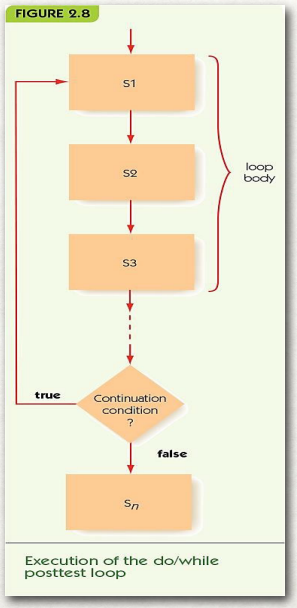
SHORTER VERSION

```
response ← "Yes"
while response = "Yes"
    get gallons used, starting mileage, ending mileage
    distance driven ← ending mileage - starting mileage
    average miles per gallon ← distance driven ÷ gallons used
    print average miles per gallon ↵
    if average miles > 25 then
        print "You are getting good gas mileage." ↵
    else
        print "You are NOT getting good gas mileage." ↵
    end if
    print "Do you want to do this again?"
    print "Enter Yes or No. "
    get response
end while
```


REPRESENTING ALGORITHMS (5 OF 5)

- Do/while, alternate iterative operation
  - Continuation condition appears at the end
  - Loop body always performed at least once
- **Primitive operations:** sequential, conditional, and iterative are all that is needed

REPRESENTING ALGORITHMS DO/WHILE POSTTEST LOOP



DISCUSSION

- Would you bother writing pseudocode?
  - why or why not?